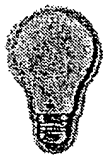
EXHIBIT A



Disclosure CA8-1999-0122

Created By: Jim Riosa
Last Modified By: Mary Fox

Required fields are marked with the asterisk (*) and must be filled in to complete the form .

Summary

Status	Sent for Search/Awaiting Results
Original Location	CA
Processing Location	END
Functional Area	15- Global Services (Div: GS) (Heald)
Submitted Date	
Owning Division	GS Add/Change
PVT Score	To calculate a PVT score, use the 'Calculate PVT' button.
Incentive Program	
Lab	
Technology Code	

Inventors with Lotus Notes IDs

Inventors: Jim Riosa/Markham/IBM, Andrew Naiberg/CanWest/IBM

Inventor Name > denotes primary contact	Inventor Serial	Div/Dept	Manager Serial	Manager Name
> Riosa, Jim	016253	IN/NJ2	000478	Maingot, Richard
Naiberg, Andrew (A.D.)	055934	15/B4D	016786	Larsson, Kath (K.E.)

Inventors without Lotus Notes IDs

IDT Selection

IDT Team:	Attorney/Patent Professional:

Response Due to IP&L :

Main Idea

***Title of disclosure (in English)**

Hierarchical Connected Graph Model For Implementation of Event Management Design in Tivoli Environments

***Idea of disclosure**

1. Describe your invention, stating the problem solved (if appropriate), and indicating the advantages of using the invention.

Event Management Design (EMD) is an IBM developed process to define the policies and procedures for managing the flow of events in a distributed computing environment. This encompasses the identification of events, filtering practices, and correlation analysis. The process is product independent. At the end of the design, the client

will be presented with a set of spreadsheets and Visio drawings for every source of events in their particular environment. The spreadsheets will contain the listings of all events, their enterprise significance, the event significance, and their necessity for correlational analysis. The Visio drawings are graphic representations of the correlations. These two documents would provide all the necessary information for someone to implement the design as a practical solution.

IBM has further extended EMD to include a Design Implementation for Tivoli (refernce discloseure SOM8-1999-0062). This disclosure essentially provides two extensions to the functionality of the tools used in EMD. There is now an additional spreadsheet that will aid in the development of BAROC files. Once certain detailed information is added to this sheet, it will automatically build the BAROC files, using whatever class names are provided. The Visio diagrams have extra code provided, that allow for the generation of TEC rules, using specified basic rule templates developed by the implementor. The new Visio code will propagate the templates with appropriate class names as determined from the drawings, as well as adding in status and severity changes, timing delays, and locations of scripts to be fired.

A further description the Visio file is required here. There is a single Visio file for each type of unique event source (hardware device, OS, application). The entire suite of correlation drawings define an Event Relationship Network (ERN). The ERN is comprised of a series of pages which show a subset of correlation relationships. Each of these pages is called a subnet. Subnets may link to other subnets, spanning as many pages as required to fully represent the set of correlational relationships. To a certain extent, the set of events on a given subnet is somewhat arbitrary. There is a physical limitation to the number of events that can be placed on any given page, and the break from page to page is the decision of the implementor. On the other hand, a subnet may contain a complete logical set of relationships, especially when it does not span to any other pages.

Problem Definition:

In a classic Tivoli implementation, one would have to consider the relationship of each event in the correlational chain with each and every other event in the chain. As such we can define a simple information function that defines the number of relationships that must be known and implemented for any given chain of events. If we let $I(fn)n$ represent the information function for a set of n related events, then a classic implementation would require the total number of states to be known to be

$$I(fn)n = n(n-1).$$

However, we can look at the event relationships not as a linear flow of events, but rather as a hierarchical connected graph. We can define these graphs as hierarchical since any event potentially implicitly contains the information of all the preceding events. If this were not the case, then we could not perform correlational analysis, as by definition, the events would share no common information, and would then be random, unrelated occurrences. By examining the set of events as a connected graph, we can reduce the amount of information required to understand an individual event's relationship to any other event in it connected network. If we let n_i be element in the chain, and p_i be the number of paths it is connected to within the graph, and we define $s = 1$ to be the sequence position along the graph, and by default a single condition of information, then the total number of conditions we need to know for any element is

$$I(fn)n_i = p_i + s$$
$$= p_i + 1$$

For a chain of x events, we can then define the total number of states to be the sum from 1 to x to be

$$\Sigma\ I(fn)n_i \text{ for } i=1 \text{ to } x$$

If we now consider a chain of events, starting with a single root cause event, which then has two branches off of it, and each branch contains 4 resultant events, we would then have the following.

For a traditional Tivoli implementation, the total number of states that would need to be known would be

$$I(fn)n = n(n-1).$$
$$= 9(8)$$
$$= 72$$

In the case of a hierarchical connected graph, we would then need to know the following number of states.

$$\Sigma\ I(fn)n_i \text{ for } i=1 \text{ to } x$$
$$= 3 + 2 + 2 + 2 + 2 + 2 + 2 + 2 + 2$$

= 21

The first element represent the root cause (2 branches + 1 positional information = 3). Since all other events only exist on a single path, they each have 1 branch + 1 position = 2.

The propagation of this path and sequence information will be done automatically by the EMD/IT tool, based on the constructs defined later in this paper.

Of the two driving purposes of TEC monitoring - correlation to determining root cause and automated response to events - only the correlation problem is being considered here.

Correlation analysis can be defined as establishing a set of relationships where the ultimate causal event can be determined. This is the primary purpose of the TEC logic engine. Automated response to a causal event is not inherently built in to the TEC code, but rather is the use of appropriate scripts executed in response to the determination of the causal event. As such, the scripts are external to the TEC code, and may vary among different implementations, depending on system management process. The development of automated scripts is a separate process driven function from the correlation analysis, and at this time will be a somewhat evolutionary process.

The Solution:

This disclosure will define the following elements:

1. A definition of events based on a connected graph model.
2. A hierarchical naming convention to be used in the BAROC (Class Definition) files.
3. Informational slot definitions required to effectively perform a correlation (root cause) analysis.
4. Basic ruleset template that handles all correlations.
5. Description of the modifications to the software toolset described in SOM8-1999-0062 to automate this process.

The Benefits:

- BAROC file class definition structure and naming structure is standardized. The naming conventions and structure are also now integrated into the logical design resulting from EMD.
- The logical flow of event correlation is totally reflected in the BAROC files. Any support person can now work through the logical structure without having access to the original EMD material.
- Path and Sequence ID searches to determine event status is more efficient than current methods. It reduces the use of all_instances, and will reduce the load on the logic engine.
- By integrating the EMD results, BAROC files and rule sets, it creates a system that lends itself to well governed change management procedures. No changes can be made on any single component without affecting the others. As such, any change will require a comprehensive review of the logical structure, and the implications for the implementation. Documentation of changes will be requisite to maintain the integrity of the integration.
- By reducing down to six essential templates, implementation can be faster, and the skill level required to implement will be reduced.
- This has the potential to be a further product that can be sold in conjunction with an EMD process for new or existing accounts.
- It will create a rulebase and BAROC structure that is consistent across the IBM Enterprise.

2. How does the invention solve the problem or achieve an advantage,(a description of "the invention", including figures inline as appropriate)?

In order to make the most efficient use of the logic engine, we will require that, at any given time, only **one** event within a given set of causal relationships will be available for correlation. All other events will have some aspect of their status changed to remove them from future correlation analysis. There are many ways this could be achieved. The event could be dropped, closed, severity changed, or a new slot could be created for all events which could be flagged for its status as a correlation candidate. The ultimate goal is that at any given time, a console could be viewed and the only events appearing would be the best estimate of a causal event at that given time.

Definition of Events:

- **Autonomous Events:** These are isolated events. In other words, they can have no causal events, nor can they be a proximate cause for any other event. There can also be no clearing events for these (see below for definition of clearing event). As such, these events will never appear in a Visio diagram, as there is no associated logic flow with their existence. All autonomous events will be handled by a single rule as defined by policy i.e. duplicate detection, escalation, trouble ticketing, etc.
- **Primary Event:** This could also be defined as the root cause event. It can have no precedent events, but may have any number of ancillary events as a result of its existence.
- **Primary/Secondary Event:** This event may be either a causal event, or may be the result of some other event proximal to the root cause event.
- **Secondary Event:** This event can only be resultant from some other event proximal to the root cause event. It will never occur spontaneously, nor can it be the causal event for any other event.
- **Clearing Event:** This event signals a return to some defined steady state or normal status. A single clearing event may clear multiple events. A clearing event may never have a causal precedent.

Two other terms must be defined here to complete our ability to describe the logical flow of events in a given system.

- **Subnet:** A subnet is the set of events with appropriate logical flow completely diagrammed. It is represented as a single page within an ERN. A subnet may be autonomous, causal, or secondary to other subnets, and by inference, other events.
- **Connector:** The connector identifies the direction of logical flow.

As such, we may now redefine the types of events as follows:

- **Autonomous Event:** has no connectors associated with it.
- **Primary Event (P):** May have *n* connectors flowing away from it; must have 0 connectors flowing into it, with the exception of a connector from a clearing event.
- **Primary/Secondary Event (P/S):** May have *n* connectors flowing in to it; may have *m* connectors flowing out of it.
- **Secondary Event (S):** May have *n* connectors flowing in to it; must have 0 connectors flowing out from it.
- **Clearing Event (C):** May have *n* connectors flowing out from it; must have 0 connectors flowing in to it.

Subnets may be defined under the same requirements. As such, you may have autonomous subnets, primary subnets, primary/secondary subnets, or secondary subnets. There is no such thing as a clearing subnet.

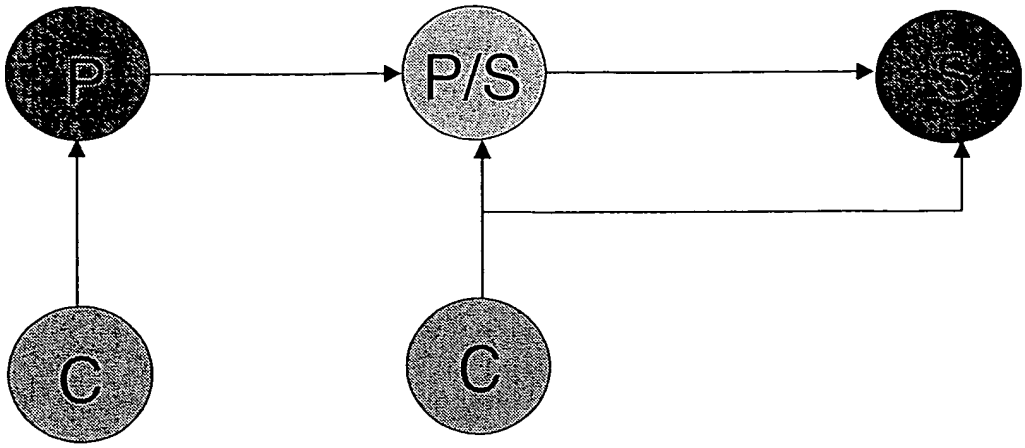
Correlation Examples:

Simple Correlations:

Single Event with Clearing Event:

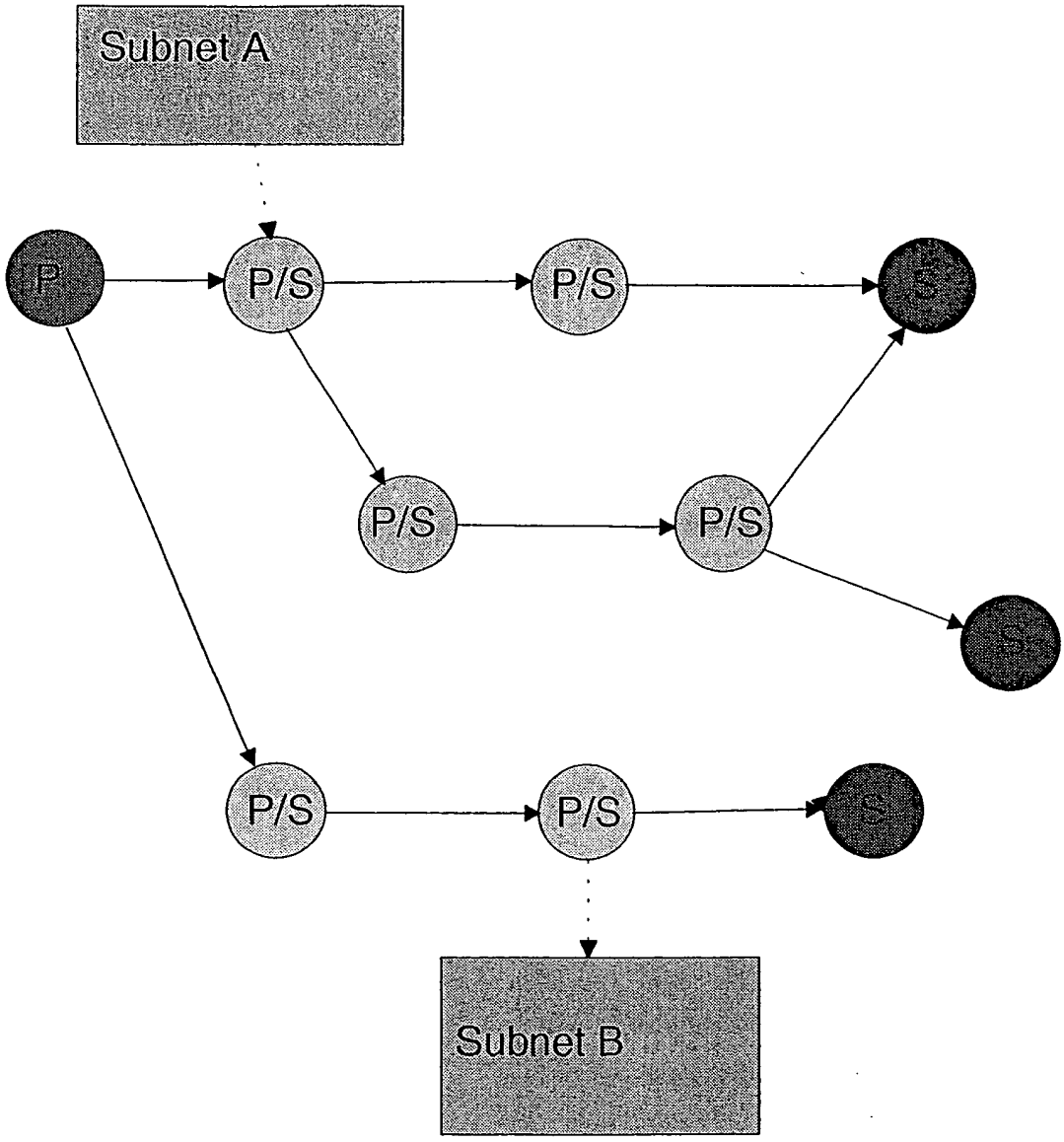
This is probably the simplest type of correlation possible. A single event occurs, and since it is not caused by any other event, nor does it cause any other event , it is by definition a primary event. The only event associated is an event that signals a return to normal events.

Single Chain of Events with Clearing Events:



This represents a more typical system of correlation, where we have a single chain of causal events, and a set of explicitly defined clearing events. It should be noted that the event that clears both the primary/secondary event and the secondary event is normally represented in Visio as only clearing the primary/secondary event, and the clearing of the secondary event is implied by the logical event flow. The explicit clearing in this case is a requirement under the automation system currently used by EMD Implementation in Tivoli. The above system better expresses the definition of a subnet, where we have a complete set of events and logical flow designed. A point to consider, which will be reintroduced later, is that only the causal chain events would be required for correlation analysis, as the clearing events are restricted to the local subnet. As such, we will need to hold in our minds the concept of a path or paths of casual events within a subnet.

Complex correlations:



This illustrates a more complex type of correlation. Intrinsic to this is the internal subnet, which consists of a single primary event, multiple primary/secondary and secondary events. Subnet A represents a casual subnet to one chain of logical flow, while Subnet B is a secondary subnet resultant from a single primary/secondary event. The logical flow illustrates the possibility of multiple paths flowing out from any particular event, as well as the possibility for multiple flow into any given event. This example provides all the necessary complexity to illustrate the proposed model for event management design proposed here.

Model for Correlation Analysis: Necessary and Sufficient Conditions

- The position of any event on a logical directed graph can be defined by two terms: path identification and sequence identification.
- Since events may belong to multiple paths, the path identification must be a list of integers. Conceptually this is the set of all paths that flow through any given event.
- The sequence identification is a single integer that defines the relative position of the event on the path or paths.

We can then redefine events in the following expressions of path ID and sequence ID

- Primary Event: Path ID ={1,2,3,...n}, Sequence ID = 0. Since the event is the root cause, it can be defined as its own class, and all other events flowing from it become a subclass of it. This accelerates the search process.
- Primary/Secondary Event: Path ID = {1,2,3,...n} Sequence ID $\in$ {n, n+1,...m}.
- Secondary Event: Path ID = {1,2,3,...n} Sequence ID = maximum integer value for any of the paths it is a member of.
- Clearing Event: Path ID = {-1, -2, -3, ...n} Sequence ID = 0. The negative path ID is required to signal that this is a clearing event, and not a candidate for root cause correlation. Implicit to this is that any event that is cleared by this event has knowledge of the negative path ID associated with it.

Therefore, the only necessary condition we require to determine the relative causal status of an event is as follows:

1. Am I in the path of a known event?
   If I am - go to step 2.
   If I am not - I am the current primary event.
2. What is my sequence ID?
   If my sequence number is greater than the known event in the path, I am then a secondary event, and will be removed from further correlational analysis.
   If my sequence number is less than the known event in the path, than I am the current primary event, and the previous known event will be removed from further correlational analysis.

It is necessary at this time to reiterate that only **one** event within a given path will remain open for correlation at any given time.

Spanning Subnets:

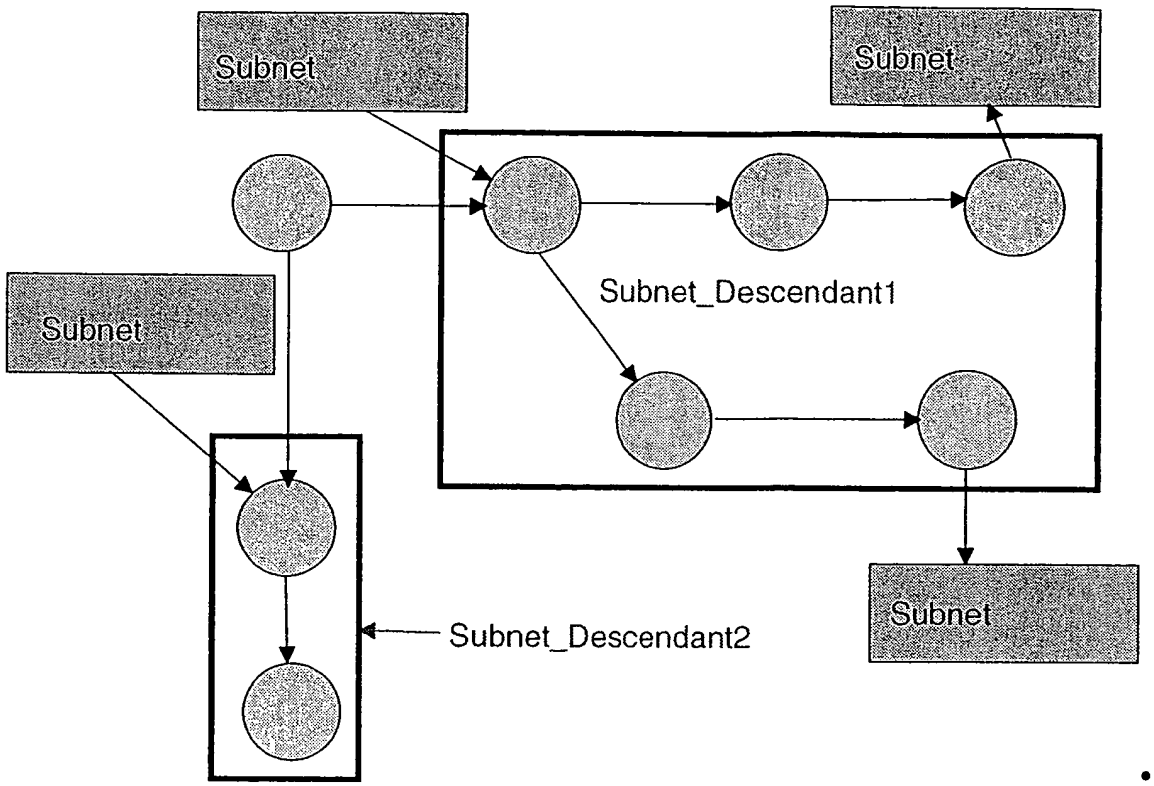
The above meta-algorithm is sufficient to determine the status (primary or secondary) of any event within a given subnet. The problem comes when we introduce multiple subnets. We could logically extend the above algorithm through all connected subnets, but the problem would become propagating all the path IDs in a consistent manner. This is not a true NP-hard problem, but is bounded by a very large polynomial time signature. However, if we were to synthesize the subnet names with the BAROC class definitions, we could minimize the search process to class structure only.

Unfortunately, simply identifying a subnet as a BAROC class is not sufficient. It is not inconceivable, and is actually probable, that only one path within a subnet leads to another subnet, and there may be other events within the primary subnet that should not be correlated with the secondary subnet. If all events in that subnet were events defined as descendant to the metaclass named for the subnet, then we would end up with spurious correlations. As such, we need to define a structure for BAROC class names and hierarchy that reflects the logical flow of events. The following is the proposed structure for BAROC class naming.

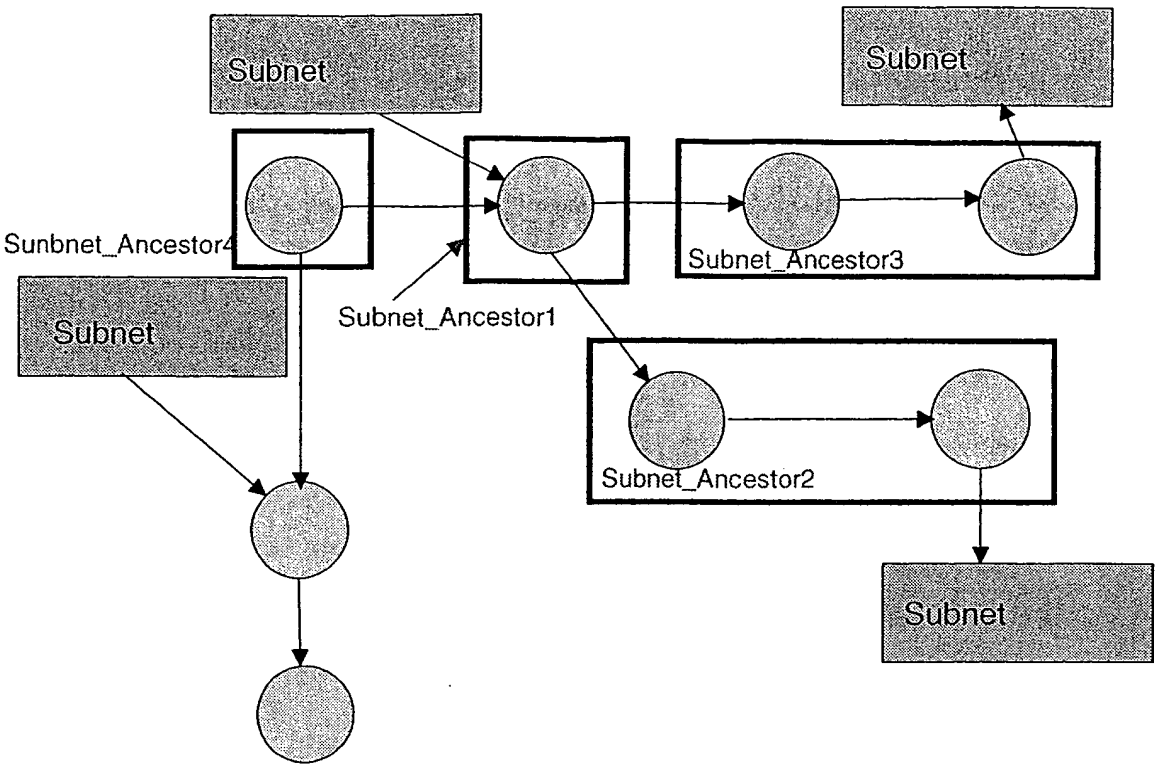
- **ERN_Class**: All events within an event source (Notes, AIX, Cisco) ill be a member of this class. Requisite slots that apply to all members of the class will be defined here (probe, probe_arg, etc.).
- **ERN_Autonomous**: All autonomous events will go in this class, since they will be handled by a single rule that handles issues like duplicate detection, trouble ticketing, etc. This is a descendant class of Event_Source_Class.
- **Subnet_Class**: All events in the subnet will be members of this class. It is also a descendant class of ERN_Class.
- **Subnet_Clearing**: All clearing events will belong in this class. This will be a descendant class of ERN_Class.

N.B. If the Subnet has no connections to other subnets, then this is the finest resolution we need to go to. All internal correlations will be dependent on path ID and sequence ID, so there will be no searching of classes beyond the Subnet_Class level. If the Subnet does connect to other subnets, then we must incorporate the following class structures.

- **Subnet_Descendant#:** If the subnet (or more correctly elements of the subnet) are secondary to events on another subnet, then all events in the direct flow from the primary subnet entry point are members of this class. It will be a descendant class of ERN_Class. The # value is merely indicative that there may be multiple flows within the subnet that need to be considered as separate classes. Consider the following example



- **Subnet_Ancestor#:** This is the logical inverse of Subnet_Descendant. However, It poses some unique problems. The general definition of this would be that if a subnet (or elements of the subnet) are primary to another subnet, then those elements will be a member of the class Subnet_Ancestor. These may be either descendant to ERN_Class or Subnet_Descendant, depending on the logical structure. The problem we face here is that unlike Subnet_Descendant, we cannot place everything in the flow into one class. The problem comes when there are branches in the logical flow that lead to multiple subnets. In order to make class naming discrete, we must use the following structure. We will define a nodal event at one that *n* paths flow out of. A proximal nodal event is one that is closer to the root cause event of the entire path, while a distal nodal event is one that is further away. The terminal event will be the event where the path flows to the next subnet, and is essentially a specialized distal nodal event. We would then cluster events into Subnet_Ancestor classes by the following simple rule. All events assigned to a specific Subnet_Ancestor will be from a distal nodal event to the next most proximal nodal event. The cluster will not include that proximal event however. Let s now reconsidered the above example in these terms.



Revision of the Correlation Template:

- The following slots will need to be defined in a BAROC file immediately descendant from root.baroc (the ancestral class definition for all events): path_id, sequence_id, descendant_class, and ancestral_class. The path_id will be a list of integers, the sequence_id is a single integer, and the descendant_class and ancestral_class will both be a list of string names.
- For any event on a subnet which has either primary or secondary subnets, and where that event lies in a logic flow that it connects to those subnets, it will have its ancestral_class slot populated with a list of all Subnet_Ancestor names for all classes which are primary to it and lie on the logical flows. The converse situation will be used for Subnet_Descendant to populate the descendant_class slot.
- Path_ID slot will be propagated with the list of all the path numbers that the event is relevant to within its own ERN. An example will be illustrated later in this document.
- Sequence_ID slot will be filled with the appropriate sequence number for that path. An example will be illustrated later.
- The generic correlation template will be as follows:

1. For any event of class within Subnet_Ancestor;

   If present, make the current event secondary, and remove from future correlational analysis.

2. For any event of class within Subnet_Descendant,

   If present, make the current event the primary event, and remove the older event from future correlational analysis.

3. For any event on the subnet (Subnet_Class) where the Path_ID of the current event intersects the Path_ID of the older event.,

   If the current event sequence_ID < older event sequence_ID, make the current event primary and remove the older event from future correlational analysis.

   If the current event sequence_ID > older event sequence_ID, remove the current event from future correlational analysis, as it is a secondary event.

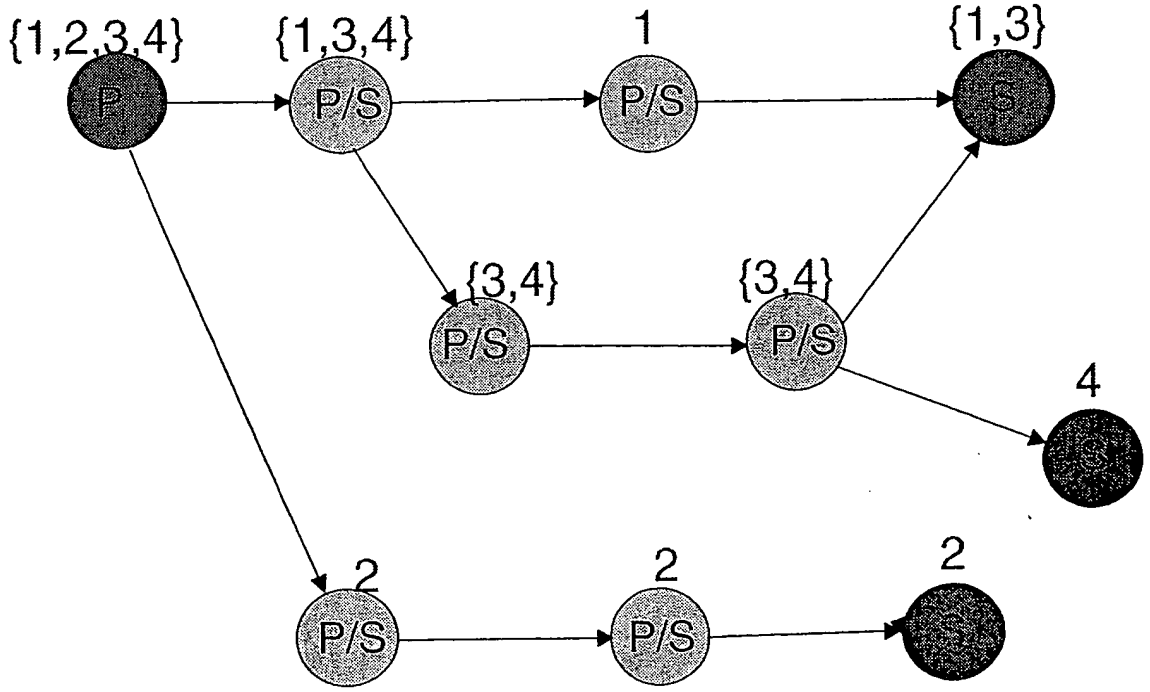
4. If all of the above conditions fail, then *de facto* this is the primary event.

- This describes the most general case for analysis. If the event had been identified in the EMD design as a true primary event, then we would remove step 1 and the second half of step 3 from the analysis, since by definition a true primary event can never be secondary to anything else. Similarly, for a true secondary event, we would remove step 2 and the first half of step 3, since a secondary event may never have any events following it.
- All templates could also include conditions for duplicate detection, trouble ticket forwarding, scripts to be fired, etc. These conditions will be determined through a final workshop with the appropriate SMEs and follow a template similar to the document in the appendix. This workshop and documentation is already a component of the current EMD Implementation in Tivoli.

Propagating Path ID numbers:

- Path ID numbers will be comprised of the set of {1, 2, *n+1...m*}.
- Starting at the root event of the ERN (either a primary or primary/secondary event) proceed to the first nodal event. The path leading in will by default be path 1. At the nodal event, one path leading out will continue to be the inherited path coming in. The remaining paths will be assigned an ascending sequence of path IDs.
- Proceed to the next nodal event on any path and repeat the procedure. **NB. No path ID number may be reused.** You must always assign the next path number in the sequence.
- Back propagate the paths. Considered a simple two branch system. The events on path 1 that are precedent to the nodal event where path 2 arises are, by definition, also members of path 2. Therefore their path_id slot must contain both numbers in its list.

Let us reconsider our complex correlation diagram.



Path ID #

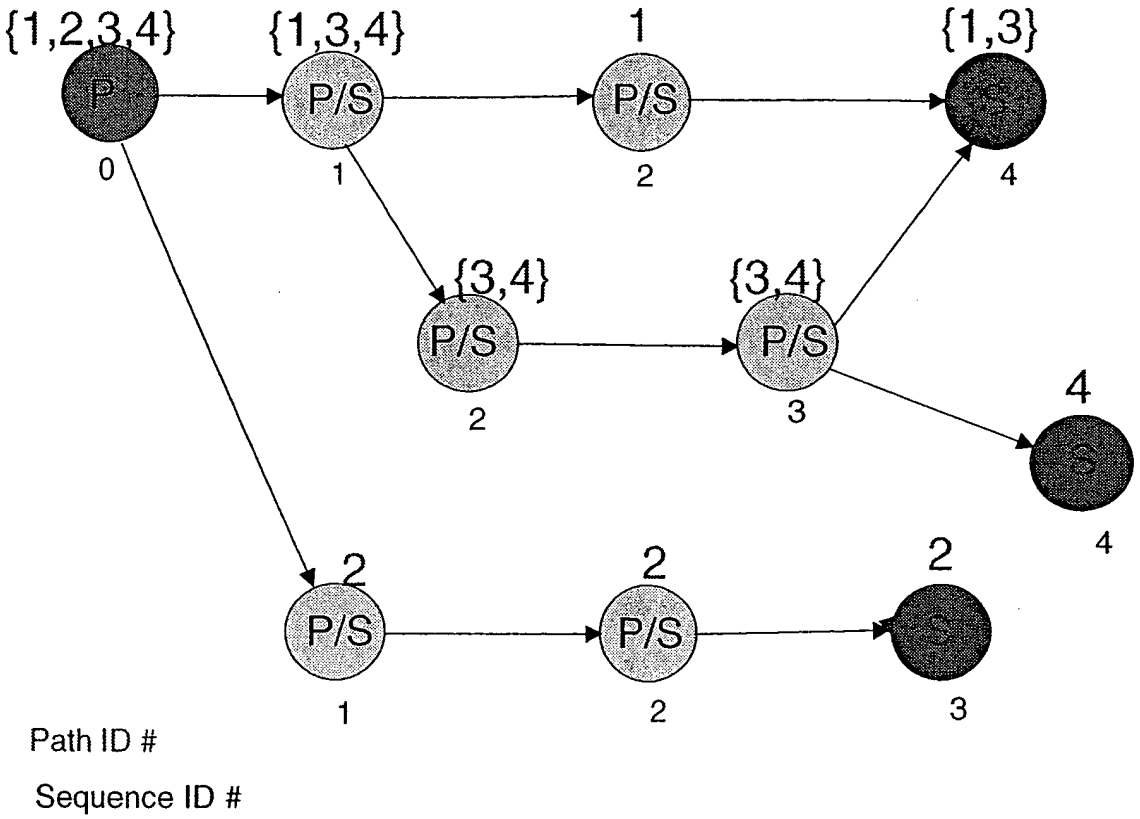
Propagating Sequence ID numbers:

- Sequence ID numbers will be comprised of the set of {1, 2, *n+1...m*}. Each event will have one unique ID number.
- Starting at the root event for the ERN, assign the sequence ID numbers down one complete path. Return to the first nodal event, and continue to assign the numbers for that path, increasing from the current sequence ID of that nodal event. For example, if the nodal event has a sequence ID of 3, then the next element down any path must

be 4. Repeat until complete.

- The only deviation is if one event is fed by two paths. At this point, that event should take the greater of the two values, and any events that follow it in the logical sequence should be increase accordingly. It is not required that the sequence ID's be a perfect *n+1* increment, only that the greater than/less than relationships be maintained along any path.

Lets us once again reconsider our complex correlation example.



Procedures for Automated Rule Generation - Extensions to the Current Methodology:

- The current methodology use the A3 and A4 sheets from the workbook to propagate a BAROC page, using the event names as class names. It separates the autonomous and correlation candidate events. This facility will need to be retained for autonomous events. The BAROC file for correlated events will be generated from the Visio drawings, propagating the appropriate slots.
- Meta-classes will need to be named as per the defined convention.
- Visio diagrams currently will verify the ERN for proper use of connectors, shapes, and subnet links. This process should be repeated.
- The Visio program will then prompt for default values related to trouble ticketing, duplicate detection, status and severity changes, time to search the event cache, and locations of scripts to be executed. These values come from the final workshop with the SMEs. It will also request the location of the templates to be used.
- The Visio program should then define the path and sequence ID numbers, and back propagate them into the appropriate parts of the BAROC file.
- The Visio program should identify primary and secondary subnets, and define the ancestor and descendant classes as appropriate, and propagate those names into the respective slots on the BAROC file.
- The Visio program will then generate the fleshed out rules based on the assigned values and templates.
- The spreadsheet will generate the BAROC files from the populated spreadsheet.

The rule set and BAROC files should be given a reality check by the implementor. If everything appears adequate, they can be loaded for testing.

A Sample Template:

The following is a template for a primary/secondary event. Elements enclosed in {} are variables that would be assigned by the SME for implementation. As such, changes in severity, status, administrators would then be automatically filled in by the tool to create a working rule.

```
rule: '{ernname}_primary_secondary_event':
(
      event: _ps_event of_class '{ernname}_Event'
            where [state: equals 'PS',
                  hostname: _hostname,
                  date: _date,
                  subnet: _subnet,
                  path_id: _path_id,
                  sequence_id: _sequence_id,
                  ancestor: _ancestor,
                  descendant: _descendant,
                  clearpath: _clearpath],
       reception_action: 'Check_Clearing_Event':
      (
            first_instance(event: _clr_event of_class '{ernname}_Event'
                  where  [subnet: equals _subnet,
                  state: equals 'C',
                  origin: equals _origin,
                  hostname: equals _hostname,
                  status: within ['ACK', 'OPEN'],
                  path_id: _clr_path_id
                  ]),
            intersect(_clearpath, _clr_path_id),
            set_event_status(_ps_event,{status1}),
            set_event_severity(_ps_event, {severity1}),
            set_event_administrapor(_ps_event, {administrator}),
/*
*           exec_prog(notification of clear),
*/
            commit_set
      ),
      reception_action: 'Check_ancestor_Subnets':
      (
            first_instance(event: _anc_event of_class within _ancestor
                  where [severity: outside ['HARMLESS'],
                        status: within ['ACK', 'OPEN']]),
            set_event_status(_ps_event, {status2}),
            set_event_severity(_ps_event, {severity1}),
            set_event_administrator(_ps_event, {administrator}),
            link_effect_to_cause( _ps_event, _anc_event),
/*
*           exec_prog(notification of secondary status),
*/
            commit_set
      ),

      action: 'Check_ERN_Relations_Secondary':
      (
            first_instance(event: _related_event of_class '{ernname}_Event'
```

```
                where [origin: equals _origin,
                hostname: equals _hostname,
                subnet: equals _subnet,
                state: within ['PS', 'P'],
                path_id: _rel_path_id,
                sequence_id: _rel_sequence_id outside [_sequence_id],
                status: within ['ACK', 'OPEN']]),

          intersect(_path_id, _rel_path_id),
          _sequence_id > _rel_sequence_id,
          set_event_status(_ps_event, {status2}),
          set_event_severity(_ps_event, {severity1}),
          set_event_administrator(_ps_event, {administrator}),
          link_effect_to_cause( _ps_event, _related_event),
/*
*         exec_prog(notification of secondary status),
*/

          commit_set
    ),
    action: 'Check_Descendant_Subnets':
    (
          first_instance(event: _desc_event of_class _within _descendant
                where [severity: outside ['HARMLESS'],
                      status: within ['ACK', 'OPEN']]),
          set_event_status(_desc_event, {status2}),
          set_event_severity(_desc_event, {severity1}),
          set_event_administrator(_desc_event, {administrator}),
          link_effect_to_cause(_desc_event, _ps_event)
/*
*         exec_prog(notification of root cause status)
*/

    ),
    action: 'update_links':
    (
          first_instance(event: _desc_event of_class within _descendant
                where [cause_date_reception: equals _date_reception,
                      cause_event_handle: equals _event_handle,
                      status: within ['ACK', 'OPEN'],
                      date_reception: _old_date_reception,
                      event_handle: _old_event_handle
                      ]),
          all_instances(event: _old_linked_events of_class 'IBMC_EVENT'
                where [cause_date_reception: equals _old_date_reception,
                      cause_event_handle: equals _old_event_handle,
                      status: within ['ACK', 'OPEN']]),
          link_effect_to_cause(_old_linked_events, _ps_event)
    ),

    action: 'Check_ERN_Relations_Primary':
    (
          first_instance(event: _related_event of_class '{ernname}_Event'
                where [origin: equals _origin,
```

```
                hostname: equals _hostname,
                subnet: equals _subnet,
                state: within ['PS', 'S'],
                path_id: _rel_path_id,
                sequence_id: _rel_sequence_id outside [_sequence_id],
                status: within ['ACK', 'OPEN']]),

          intersect(_path_id, _rel_path_id),
          _sequence_id < _rel_sequence_id,
          set_event_status(_related_event, {status2}),
          set_event_severity(_related_event, {severity1}),
          set_event_administrator(_related_event, {administrator}),
          link_effect_to_cause(_related_event, _ps_event)
/*
*         exec_prog(notification of root cause status)
*/
    ),

    action: 'update_links2':
    (
          first_instance(event: _related_event of_class '{ernname}_Event'
                where [cause_date_reception: equals _date_reception,
                      cause_event_handle: equals _event_handle,
                      status: within ['ACK', 'OPEN'],
                      date_reception: _old_date_reception,
                      event_handle: _old_event_handle
                      ]),
          all_instances(event: _old_linked_events of_class 'IBMC_EVENT'
                where [cause_date_reception: equals _old_date_reception,
                      cause_event_handle: equals _old_event_handle,
                      status: within ['ACK', 'OPEN']]),
          link_effect_to_cause(_old_linked_events, _ps_event)
    ),

    action:'notify_on_primary':
    (
/*
*         exec_prog(notification of root cause status),
*/
          commit_set
    )

).
```

Obviously, there are any number of actions that could be taken at different levels. This is only one possible example, and is set by the policy workshop with the SMEs. A further implication is that the rules for primary events would contain only the lower half of this rule, while secondary events would contain the upper half. In a uniform environment, where all event sources are handled in an identical fashion, we would end up with a total of 6 rule sets: duplicate detection and trouble ticketing; autonomous events; primary events; primary/secondary events; secondary events; clearing events.

3. If the same advantage or problem has been identified by others (inside/outside IBM), how have those others solved it and does your solution differ and why is it better?

The current state of the EMD-IT tool, while automating certain aspects of implementing an Event Management Design, has two limitations. The implementor is still required to create the ruleset templates to handle correlation, and will generate a rule for every type of event (primary, primary/secondary, secondary, and clearing event) for each subnet within the ERN. As such, there are a large number of rules generated, and the implementor is still required to develop the bulk of the code to do correlation. Also, while there is facility to generate BAROC files, this is a manual procedure to establish any hierarchical information.

This methodolgy would provide a template with all the sufficient code to handle correlational analysis. The implementor would only need to add specific actions after the correlation was complete. These actions are comparatively simple and require little training to implement. Because of the nature of this type of correaltion, one only requires 4 rules for the entire ERN, rather than 4 for each rule on the ERN.

Since the structure of the BAROC files in a well mannered hierarchy is requisite for this model, BAROC file generation is now completely automated.

4. If the invention is implemented in a product or prototype, include technical details, purpose, disclosure details to others and the date of that implementation.

This methodology has been implemented for the Tivoli server monitoring for IBMC. EMD's were performed for the following services, Notes, Risc/AIX/SP2, ADSM, PSM and Print, and LAN Network. The methodology and modified tool were used to generate the majority of the rules to implement the results of these EMDs on a single Tivoli TEC. Deployment of the rulebase was

Attached are a sample of the BAROC files and rulesets for the implementation. This represents the complete results for the Notes Service.


Notes_Auto.baroc


notes.baroc


ibmc.baroc


notes_s.rls


notes_c.rls


notes_dup.rls


notes_p.rls


notes_ps.rls


auto_notify.rls

***Critical Questions (Questions 1 - 7 must be answered)**

***Question 1**
On what date was the invention workable? **Please format the date as MM/DD/YYYY**
(Workable means i.e. when you know that your design will solve the problem)

***Question 2** Is there any planned or actual publication or disclosure of your invention to anyone outside IBM?	○ Yes ● No
If yes, Enter the name of each publication or patent and the date published below. Publication/Patent: Date Published or Issued:	
Are you aware of any publications, products or patents that relate to this invention?	● Yes ○ No
If yes, Enter the name of each publication or patent and the date published below. Publication/Patent: SOM8-1999-0062 Date Published or Issued:	

***Question 3** Has the subject matter of the invention or a product incorporating the invention been sold, used internally in manufacturing, announced for sale, or included in a proposal?	○ Yes ● No
Is a sale, use in manufacturing, product announcement, or proposal planned?	○ Yes ● No
If Yes, identify the product if known and indicate the date or planned date of sale, announcements, or proposal and to whom the sale, announcement or proposal has been or will be made. Product: Version/Release: Code Name: Date: To Whom: If more than one, use cut and paste and append as necessary in the field provided.	

***Question 4** Was the subject matter of your invention or a product incorporating your invention used in public, e.g., outside IBM or in the presence of non-IBMers?	○ Yes ● No
If yes, give a date. **Please format the date as MM/DD/YYYY**	

***Question 5** Have you ever discussed your invention with others not employed at IBM?	○ Yes ● No
If yes, identify individuals and date discussed. Fill in the text area with the following information, the names of the individuals, the employer, date discussed, under CDA, and CDA #.	

***Question 6** Was the invention, in any way, started or developed under a government contract or project?	○ Yes ● No ○ Not sure
If Yes, enter the contract number	

***Question 7** Was the invention made in the course of any alliance, joint development or other contract activities?	○ Yes ● No ○ Not Sure
If Yes, enter the following :Name of Alliance, Contractor or Joint Developer	
Contract ID number	
Relationship contact name	

Relationship contact E-mail
Relationship contact phone

Question 8 Have you submitted, or are you aware of, any related disclosure submission?	○ Yes ● No
If Yes, please provide the title and docket or disclosure number below:	

Question 9

What type of companies do you expect to compete with inventions of this type? *Check all that apply.*

- ☐ Manufacturers of enterprise servers
- ☐ Manufacturers of entry servers
- ☐ Manufacturers of workstations
- ☐ Manufacturers of PC's
- ☐ Non-computer manufacturers
- ☐ Developers of operating systems
- ☐ Developers of networking software
- ☒ Developers of application software
- ☒ Integrated solution providers
- ☒ Service providers
- ☐ Other (Please specify below)

Patent Value Tool (Optional - this may be used by the inventor and attorney to assist with the evalua

(The Patent Value tool can be used by you or the evaluation team to determine the potential licensing value of your invention.)

The **Patent Value Tool** has not yet been used to calculate a score.

Evaluation

This evaluation was entered by **Mary Fox/Markham/IBM** on
Team Evaluation What is the team's evaluation of this disclosure? Search
Date rated :
Evaluation Comments

Search Information

Date sent:	Target completion date: 03/17/2000	Actual completion date:
Who was the search sent to (This area is to designate a Local Searcher name or WAIPL):		
Send search request to: Patricia Boykin/Arlington/IBM@IBMUS Abdi Dirie/Arlington/IBM@IBMUS	Search Type: ☒ Patentability ☒ Clearance ☐ Validity ☐ State of Art	
Features to be searched: As per invention disclosure		

Post Disclosure Text & Drawings

Enter any additional information relating to this disclosure below:

(Form Revised)